Foundation of Cryptography (0368-4162-01), Lecture 7

MACs and Signatures

Iftach Haitner, Tel Aviv University

December 27, 2011

Section 1

Message Authentication Code (MAC)

Message Authentication Code (MAC)

Goal: message authentication.

Message Authentication Code (MAC)

Goal: message authentication.

Definition 1 (MAC)

A trippet of PPT's (Gen, Mac, Vrfy) such that

1. $\mathsf{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathsf{Mac}(k, m)$ outputs a "tag" t
3. $\mathsf{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

Message Authentication Code (MAC)

Goal: message authentication.

Definition 1 (MAC)

A trippet of PPT's $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ such that

1. $\mathrm{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathrm{Mac}(k, m)$ outputs a "tag" t
3. $\mathrm{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(k, m, t) = 1$ for any $k \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^n$ and $t = \mathrm{Mac}(k, m)$

Message Authentication Code (MAC)

Goal: message authentication.

Definition 1 (MAC)

A trippet of PPT's $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ such that

1. $\mathrm{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathrm{Mac}(k, m)$ outputs a "tag" t
3. $\mathrm{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(k, m, t) = 1$ for any $k \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^n$ and $t = \mathrm{Mac}(k, m)$
- **Unforgability:** For any oracle-aided PPT A:

$$\Pr\left[k \leftarrow \mathrm{Gen}(1^n); (m,t) \leftarrow \mathrm{A}^{\mathrm{Mac}_k, \mathrm{Vrfy}_k}(1^n) : \mathrm{Vrfy}_k(m,t) = 1 \wedge \mathrm{Mac}_k \text{ was not asked on } m\right] \leq \mathrm{neg}(n)$$

- “Private key" definition

- “Private key" definition
- Definition too strong?

- “Private key" definition
- Definition too strong? Any message? Use of Verifier?

- “Private key" definition
- Definition too strong? Any message? Use of Verifier?
- “Reply attacks"

- “Private key" definition
- Definition too strong? Any message? Use of Verifier?
- “Reply attacks"
- Will focus on bounded length messages (specifically n), and then show how to move to any length

Bounded MACs

Definition 2 (ℓ-time MAC)

Same as in Definition 1, but security is only required against ℓ-query adversaries.

One-time MAC

Construction 3 (One-time MAC)

- $\mathrm{Gen}(1^n)$: outputs $k \leftarrow \{0,1\}^n$
- $\mathrm{Mac}(k, m) = k \oplus m$
- $\mathrm{Vrfy}(k, m, t) = 1$, iff $t = k \oplus m$

ℓ-times MAC

Construction 4 (ℓ-time MAC, Stateful)

Like Construction 3, but uses a different mask for every message.

ℓ-times MAC

Construction 4 (ℓ-time MAC, Stateful)

Like Construction 3, but uses a different mask for every message.

Construction 5 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient family of ℓ-wise independent hash functions.[a]

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h, m) = h(m)$
- $\mathsf{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

[a] For any distinct $x_1, \ldots, x_\ell \in \{0,1\}^n$ and $y_1, \ldots, y_\ell \in \{0,1\}^n$, $\Pr_{h \leftarrow \mathcal{H}_n}[h(x_1) = y_1 \wedge \cdots \wedge h(x_\ell) = y_\ell] = 2^{-tn}$.

OWF $\implies$ MAC

Construction 6 (PRF-based MAC)

Same as Construction 5, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

OWF $\Longrightarrow$ MAC

Construction 6 (PRF-based MAC)

Same as Construction 5, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 7

Assuming that $\mathcal{F}$ is a PRF, then Construction 6 is a MAC.

Proof:

OWF $\implies$ MAC

Construction 6 (PRF-based MAC)

Same as Construction 5, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 7

Assuming that $\mathcal{F}$ is a PRF, then Construction 6 is a MAC.

Proof: Easy to prove if $\mathcal{F}$ is a family of random functions. Hence, also holds in case $\mathcal{F}$ is a PRF. $\square$

Length restricted MAC $\Longrightarrow$ MAC

Construction 8 (Length restricted MAC $\Longrightarrow$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be length restricted MAC with $d(n) = n$, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an eff. function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Length restricted MAC $\Longrightarrow$ MAC

Construction 8 (Length restricted MAC $\Longrightarrow$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be length restricted MAC with $d(n) = n$, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an eff. function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Claim 9

Assume $\mathcal{H}$ is “collision resistant", then $(\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ is a MAC.

Length restricted MAC $\Longrightarrow$ MAC

Construction 8 (Length restricted MAC $\Longrightarrow$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be length restricted MAC with $d(n) = n$, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an eff. function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Claim 9

Assume $\mathcal{H}$ is “collision resistant", then $(\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ is a MAC.

Collision Resistant Hash Family

Definition 10 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \\ \wedge h(x) = h(x')] \leq \operatorname{neg}(n)$$

for any PPT A.

Collision Resistant Hash Family

Definition 10 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \wedge h(x) = h(x')] \leq \mathrm{neg}(n)$$

for any PPT A.

- Not known to be implied by OWF

Collision Resistant Hash Family

Definition 10 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \\ \wedge h(x) = h(x')] \leq \operatorname{neg}(n)$$

for any PPT A.

- Not known to be implied by OWF

Proof: (of Claim 9) HW

Section 2

Signature Schemes

Definition

Definition 11 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathrm{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$ (we let $\mathrm{Sign}_s(\cdot) := \mathrm{Sign}(s, \cdot)$)
3. $\mathrm{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO) (we let $\mathrm{Vrfy}_v(\cdot) := \mathrm{Vrfy}(v, \cdot)$)

Definition

Definition 11 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathsf{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathsf{Sign}(s, m)$ outputs a “signature" $\sigma \in \{0,1\}^*$ (we let $\mathsf{Sign}_s(\cdot) := \mathsf{Sign}(s, \cdot)$)
3. $\mathsf{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO) (we let $\mathsf{Vrfy}_v(\cdot) := \mathsf{Vrfy}(v, \cdot)$)

- **Consistency:** $\mathsf{Vrfy}_v(m, \sigma) = 1$ for any $(s, v) \in \mathsf{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathsf{Supp}(\mathsf{Sign}_s(m))$

Definition

Definition 11 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathsf{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathsf{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$ (we let $\mathsf{Sign}_s(\cdot) := \mathsf{Sign}(s, \cdot)$)
3. $\mathsf{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO) (we let $\mathsf{Vrfy}_v(\cdot) := \mathsf{Vrfy}(v, \cdot)$)

- **Consistency:** $\mathsf{Vrfy}_v(m, \sigma) = 1$ for any $(s, v) \in \mathrm{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathrm{Supp}(\mathsf{Sign}_s(m))$
- **Unforgability:** For any oracle-aided PPT A

$$\Pr\left[(s, v) \leftarrow \mathsf{Gen}(1^n); (m, \sigma) \leftarrow \mathsf{A}^{\mathsf{Sign}_s(1^n, v)} : \right.$$
$$\left. \mathsf{Vrfy}_v(m, \sigma) = 1 \wedge \mathsf{Sign}_s \text{ was not asked on } m\right] \leq \mathrm{neg}(n)$$

- “Harder" to construct than MACs: (even restricted forms) require OWF

- “Harder" to construct than MACs: (even restricted forms) require OWF
- A can emulate oracle access to Vrfy by itself

- “Harder" to construct than MACs: (even restricted forms) require OWF
- A can emulate oracle access to Vrfy by itself
- Strong Signatures: impossible to generate new valid signatures

Section 3

OWF $\implies$ Signature

One Time Signatures

Definition 12 (one time signatures)

Same as Definition 11, but A can only ask for a single signature.

One Time Signatures

Definition 12 (one time signatures)

Same as Definition 11, but A can only ask for a single signature.

Definition 13 (length restricted, one time signatures)

Same as Definition 26, but $A(1^n)$ can only for signatures of predetermined length (in our case n).

One Time Signatures

Definition 12 (one time signatures)

Same as Definition 11, but A can only ask for a single signature.

Definition 13 (length restricted, one time signatures)

Same as Definition 26, but $A(1^n)$ can only for signatures of predetermined length (in our case n).

- Assuming CRH exists: length restricted, one time signatures $\Longrightarrow$ one time signatures.

OWF $\implies$ length restricted, One Time Signature

Construction 14 (length restricted, one time signature)

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$.

1. Gen(1^n): $s_1^0, s_1^1, \ldots, s_n^0, s_n^1 \leftarrow \{0,1\}^n$, let $s = (s_1^0, s_1^1, \ldots, s_n^0, s_n^1)$ and $v = (v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \ldots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1))$
2. Sign(s, m): Output $(s_1^{m_1}, \ldots, s_n^{m_n})$
3. Vrfy$(v, m, \sigma = (\sigma_1, \ldots, \sigma_n))$ check that $f(\sigma_i) = v_{m_i}$ for all $i \in [n]$

OWF $\implies$ length restricted, One Time Signature

Construction 14 (length restricted, one time signature)

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$.

1. $\mathrm{Gen}(1^n)$: $s_1^0, s_1^1, \ldots, s_n^0, s_n^1 \leftarrow \{0,1\}^n$, let $s = (s_1^0, s_1^1, \ldots, s_n^0, s_n^1)$ and $v = (v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \ldots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1))$
2. $\mathrm{Sign}(s, m)$: Output $(s_1^{m_1}, \ldots, s_n^{m_n})$
3. $\mathrm{Vrfy}(v, m, \sigma = (\sigma_1, \ldots, \sigma_n))$ check that $f(\sigma_i) = v_{m_i}$ for all $i \in [n]$

Lemma 15

Assume that f is a OWF, then scheme from Construction 14 is a length restricted one-time signature scheme

Proving Lemma 15

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 14, we use A to invert f.

Algorithm 16

(Inv) **Input:** $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$ with y.
2. If $A(1^n, v)$ ask to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

Proving Lemma 15

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 14, we use A to invert f.

Algorithm 16

(Inv) **Input:** $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v^{i^*}_{j^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$ with y.
2. If $A(1^n, v)$ ask to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real “signature game" (ind. of i^*).

Proving Lemma 15

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 14, we use A to invert f.

Algorithm 16

(Inv) **Input:** $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$ with y.
2. If $A(1^n, v)$ ask to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real "signature game" (ind. of i^*).
Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for any $n \in \mathcal{I}$.

Proving Lemma 15

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 14, we use A to invert f.

Algorithm 16

(Inv) **Input:** $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v^{i^*}_{j^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$ with y.
2. If $A(1^n, v)$ ask to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real “signature game" (ind. of i^*).
Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for any $n \in \mathcal{I}$.

- Non length-restricted one-time signatures?

Proving Lemma 15

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 14, we use A to invert f.

Algorithm 16

(Inv) **Input:** $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$ with y.
2. If $A(1^n, v)$ ask to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real “signature game" (ind. of i^*). Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for any $n \in \mathcal{I}$.

- Non length-restricted one-time signatures? use CRH

Proving Lemma 15

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 14, we use A to invert f.

Algorithm 16

(Inv) **Input:** $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$ with y.
2. If $A(1^n, v)$ ask to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real "signature game" (ind. of i^*).
Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for any $n \in \mathcal{I}$.

- Non length-restricted one-time signatures? use CRH ??

Stateful schemes (also known as, Memory-dependant schemes)

Definition 17 (Stateful scheme)

Same as in Definition 11, but Sign might keep state.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 17 (Stateful scheme)

Same as in Definition 11, but Sign might keep state.

- Make sense in many applications (e.g., , smartcards)

Stateful schemes (also known as, Memory-dependant schemes)

Definition 17 (Stateful scheme)

Same as in Definition 11, but Sign might keep state.

- Make sense in many applications (e.g., , smartcards)
- We'll use it a building block for building a stateless scheme

Naive construction

Let (Gen, Sign, Vrfy) be a one-time signature scheme.

Construction 18 (Naive construction)

1. $\mathrm{Gen}'(1^n)$ outputs $(s, v) = \mathrm{Gen}(1^n)$.
2. $\mathrm{Sign}_s(m_i)$, where m_i is i'th message to sign:
   Let $((m_1, \sigma_1'), \ldots, (m_{i-1}, \sigma_{i-1}'))$ be the previously signed pairs of messages/signatures.
   1. Let $(s_i, v_i) \leftarrow \mathrm{Gen}(1^n)$
   2. Let $\sigma_i = \mathrm{Sign}_{s_{i-1}}(m_i, v_i)$, where $s_0 = s$, and output $\sigma_i' = (\sigma_{i-1}', m_i, v_i, \sigma_i)$.
3. $\mathrm{Vrfy}_v'(m, \sigma' = (m_1, v_1, \sigma_1), \ldots, (m_i, v_i, \sigma_i))$:
   1. Check that $m_i = m$.
   2. $\forall j \in [i]$, verify that $\mathrm{Vrfy}_{v_{i-1}}((m_i, v_i), \sigma_i) = 1$, where $v_0 = v$.

Stateful schemes

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures
3. Critically uses the fact that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is works for any length

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures
3. Critically uses the fact that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is works for any length

Lemma 19

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures
3. Critically uses the fact that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is works for any length

Lemma 19

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

Proof: Let a PPT A', $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that breaks the security of $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$, we present a PPT A that breaks the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures
3. Critically uses the fact that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is works for any length

Lemma 19

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

Proof: Let a PPT A', $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that breaks the security of $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$, we present a PPT A that breaks the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.
We assume for simplicity that p also bounds the query complexity of A'

Proving Lemma 19 cont.

Let (the r.v) $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q))$ be the pair output by A'.

Proving Lemma 19 cont.

Let (the r.v) $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q))$ be the pair output by A'.

Claim 20

Whenever A' breaks the scheme, $\exists i \in [q]$ s.t. :

1. Sign$'$ *was not* asked by A' on m_i.
2. Sign$'$ *was* asked by A' on $m_{i'}$, for every $i' \in [i-1]$

Proving Lemma 19 cont.

Let (the r.v) $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q))$ be the pair output by A′.

Claim 20

Whenever A' breaks the scheme, $\exists i \in [q]$ s.t. :

1. Sign′ *was not* asked by A' on m_i.
2. Sign′ *was* asked by A' on $m_{i'}$, for every $i' \in [i-1]$

Proof:

Proving Lemma 19 cont.

Let (the r.v) $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q))$ be the pair output by A′.

Claim 20

Whenever A' breaks the scheme, $\exists i \in [q]$ s.t. :

1. Sign′ *was not* asked by A' on m_i.
2. Sign′ *was* asked by A' on $m_{i'}$, for every $i' \in [i-1]$

Proof: Let i be the maximal index such that condition (2) holds (cannot be q).$\square$

Proving Lemma 19 cont.

Let (the r.v) $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q))$ be the pair output by A′.

Claim 20

Whenever A' breaks the scheme, $\exists i \in [q]$ s.t. :

1. Sign′ *was not* asked by A' on m_i.
2. Sign′ *was* asked by A' on $m_{i'}$, for every $i' \in [i-1]$

Proof: Let i be the maximal index such that condition (2) holds (cannot be q).$\square$

Let i be the index guaranteed by Claim 20 in a successful attack of A′.

Proving Lemma 19 cont.

Let (the r.v) $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q))$ be the pair output by A′.

Claim 20

Whenever A' breaks the scheme, $\exists i \in [q]$ s.t. :

1. Sign′ *was not* asked by A' on m_i.
2. Sign′ *was* asked by A' on $m_{i'}$, for every $i' \in [i-1]$

Proof: Let i be the maximal index such that condition (2) holds (cannot be q).$\square$

Let i be the index guaranteed by Claim 20 in a successful attack of A′.

Hence, $\mathrm{Sign}_{s_i}(\sigma_i, m_i^* = (m_i, v_i)) = 1$, where s_i is the signing key generated by Sign'_s when signing m_{i-1}, and Sign_{s_i} was not queried (by Sign'_s) on m_i^*.

Definition of A

We define algorithm A as follows:

Algorithm 21 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathrm{Gen}'(1^n)$.
2. Emulate a random execution of $\mathrm{A}'^{\mathrm{Sign}'_{s'}}$ with a single twist:
   - On the i'th call to $\mathrm{Sign}'_{s'}$, set $v_i = v$ (rather then choosing it via Gen)
   - When need to sign using s_i, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \mathrm{A}'$
4. Output $(m_i^* = (m_i, v_i), \sigma_i)$ (aborts, if $i > q$))

Definition of A

We define algorithm A as follows:

Algorithm 21 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
2. Emulate a random execution of $\text{A}'^{\text{Sign}'_{s'}}$ with a single twist:
   - On the i'th call to $\text{Sign}'_{s'}$, set $v_i = v$ (rather then choosing it via Gen)
   - When need to sign using s_i, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \text{A}'$
4. Output $(m^*_i = (m_i, v_i), \sigma_i)$ (aborts, if $i > q$))

- Sign_s is called at most once

Definition of A

We define algorithm A as follows:

Algorithm 21 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathrm{Gen}'(1^n)$.
2. Emulate a random execution of $\mathrm{A}'^{\mathrm{Sign}'_{s'}}$ with a single twist:
   - On the i'th call to $\mathrm{Sign}'_{s'}$, set $v_i = v$ (rather then choosing it via Gen)
   - When need to sign using s_i, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \mathrm{A}'$
4. Output $(m_i^* = (m_i, v_i), \sigma_i)$ (aborts, if $i > q$))

- Sign_s is called at most once
- The emulated game $\mathrm{A}'^{\mathrm{Sign}'_{s'}}$ has the “right" distribution.

Analysis of A

Let $i(m, \sigma)$ be the index guaranteed by Claim 20 (set to $\perp$, if A′ dos not break the scheme).

Analysis of A

Let $i(m, \sigma)$ be the index guaranteed by Claim 20 (set to $\perp$, if A$'$ dos not break the scheme).

- A breaks $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ whenever $i = i(m, \sigma)$.

Analysis of A

Let $i(m,\sigma)$ be the index guaranteed by Claim 20 (set to $\perp$, if A$'$ dos not break the scheme).

- A breaks $(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})$ whenever $i = i(m,\sigma)$.

Hence, for any $i \in \mathcal{I}$

$$\begin{aligned}
&\Pr[\mathsf{A} \text{ breaks } (\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})]\\
&\quad \geq \quad \Pr_{i \leftarrow [p = p(n)]}[i = i(m,\sigma)]\\
&\quad \geq \quad \frac{1}{p} \cdot \Pr[\mathsf{A}' \text{ breaks } (\mathsf{Gen}',\mathsf{Sign}',\mathsf{Vrfy}')] \geq \frac{1}{p(n)^2}
\end{aligned}$$

"Semi"-Stateful Schemes

A one-time scheme (Gen, Sign, Vrfy), and $\ell = \ell(n) \in \omega(\log n)$

Construction 22

- $\mathrm{Gen}'(1^n)$: output $(s, v) \leftarrow \mathrm{Gen}(1^n)$.
- $\mathrm{Sign}_s(m)$: choose $r = (r_1 \ldots, r_\ell) \leftarrow \{0,1\}^\ell$ and let $(s_\lambda, v_\lambda) = (s, v)$
  1. For $i = 0$ to $\ell - 1$: if $a_{r_{1,\ldots,i}}$ was not set:
     1. For $j \in \{0,1\}$, let $(s_{r_{1,\ldots,i},j}, v_{r_{1,\ldots,i},j}) \leftarrow \mathrm{Gen}(1^n)$
     2. $\sigma_{r_{1,\ldots,i}} = \mathrm{Sign}_{s_{r_{1,\ldots,i}}}(v_{r_{1,\ldots,i},0}, v_{r_{1,\ldots,i},1})$
     3. $a_{r_{1,\ldots,i}} = (v_{r_{1,\ldots,i},0}, v_{r_{1,\ldots,i},1}, \sigma_{r_{1,\ldots,i}})$
  2. Output $(r, a_\lambda, a_{r_1}, \ldots, a_r, \sigma = \mathrm{Sign}_{s_r}(m))$
- $\mathrm{Vrfy}'_v(m, \sigma' = (r, a_\lambda, a_{r_1}, \ldots, a_r, \sigma)$:
  1. For every $i \in [\ell]$, verify that $\mathrm{Vrfy}_{v_{r_{1,\ldots,i-1}}}(a_{r_{1,\ldots,i}}) = 1$.
  2. Verify that $\mathrm{Vrfy}_{v_r}(m, \sigma) = 1$

1. A much more efficient scheme

1. A much more efficient scheme
2. With save but negligible probability, each leaf is used once.

1. A much more efficient scheme
2. With save but negligible probability, each leaf is used once.
3. We sign both descendants, to avoid using the same one-time signature twice

1. A much more efficient scheme
2. With save but negligible probability, each leaf is used once.
3. We sign both descendants, to avoid using the same one-time signature twice
4. Sign$'$ does not keep track of the messages it signed

1. A much more efficient scheme
2. With save but negligible probability, each leaf is used once.
3. We sign both descendants, to avoid using the same one-time signature twice
4. Sign$'$ does not keep track of the messages it signed
5. State is only needed to maintain the secret keys and to avoid using the same one-time signature twice.

1. A much more efficient scheme
2. With save but negligible probability, each leaf is used once.
3. We sign both descendants, to avoid using the same one-time signature twice
4. Sign' does not keep track of the messages it signed
5. State is only needed to maintain the secret keys and to avoid using the same one-time signature twice.

Lemma 23

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

1. A much more efficient scheme
2. With save but negligible probability, each leaf is used once.
3. We sign both descendants, to avoid using the same one-time signature twice
4. Sign' does not keep track of the messages it signed
5. State is only needed to maintain the secret keys and to avoid using the same one-time signature twice.

Lemma 23

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

Proof: ?

Stateless Scheme

Inefficient scheme:
Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \mathrm{poly}$ is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$

Stateless Scheme

Inefficient scheme:
Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$: when setting $(s_{r_{1,\ldots,i},j}, v_{r_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(r_{1,\ldots,i},j)$ as the randomness for Gen.

Stateless Scheme

Inefficient scheme:
Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$: when setting $(s_{r_{1,\ldots,i},j}, v_{r_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(r_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign$'$ keeps no state

Stateless Scheme

Inefficient scheme:
Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$: when setting $(s_{r_{1,\ldots,i},j}, v_{r_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(r_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign' keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$: when setting $(s_{r_{1,\ldots,i},j}, v_{r_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(r_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign$'$ keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Efficient scheme:

Stateless Scheme

Inefficient scheme:
Let $\Pi_{\ell,q}$ be the set of random functions from $\bigcup_{i\in[\ell]}\{0,1\}^i$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v)\leftarrow \mathsf{Gen}(1^n)$ and $\pi\leftarrow\Pi_{\ell(n),q(n)}$, where $q\in$ poly is large enough for the application below, and outputs $(s'=(s,\pi), v'=v)$
2. $\mathsf{Sign}'(1^n)$: when setting $(s_{r_{1,\dots,i},j}, v_{r_{1,\dots,i},j})\leftarrow \mathsf{Gen}(1^n)$, use $\pi(r_{1,\dots,i},j)$ as the randomness for Gen.

- Sign$'$ keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Efficient scheme: use PRF

Without CRH

Definition 24 (target collision resistant (TCR))

An function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if for any PPT A: $\Pr[x \leftarrow A(1^n), h \leftarrow \mathcal{H}_n : x' \leftarrow A(x, h) : x \neq x' \wedge h(x) = h(x')] \leq \mathrm{neg}(n)$

Without CRH

Definition 24 (target collision resistant (TCR))

An function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if for any PPT A: $\Pr[x \leftarrow A(1^n), h \leftarrow \mathcal{H}_n : x' \leftarrow A(x, h) : x \neq x' \wedge h(x) = h(x')] \leq \operatorname{neg}(n)$

Theorem 25

OWFs imply TCR.

Definition 26 (target one-time signatures)

Same as one time signature, but A has to declare its query *before* seing the verification key.

Definition 26 (target one-time signatures)

Same as one time signature, but A has to declare its query *before* seing the verification key.

Claim 27

OWFs imply target one-time signatures

Definition 26 (target one-time signatures)

Same as one time signature, but A has to declare its query *before* seing the verification key.

Claim 27

OWFs imply target one-time signatures

Lemma 28

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a target one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *from Construction 22 is a stateful signature scheme.*

Definition 26 (target one-time signatures)

Same as one time signature, but A has to declare its query *before* seing the verification key.

Claim 27

OWFs imply target one-time signatures

Lemma 28

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a target one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *from Construction 22 is a stateful signature scheme.*

Proof: ?

Definition 26 (target one-time signatures)

Same as one time signature, but A has to declare its query *before* seing the verification key.

Claim 27

OWFs imply target one-time signatures

Lemma 28

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a target one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *from Construction 22 is a stateful signature scheme.*

Proof: ?

- Reduction to stateless scheme as in the CRH based scheme